Part I

Elliptic curves cryptography and factorization

EMPIRICAL NOTION of SECRECY of CRYPTOSYSTEMS

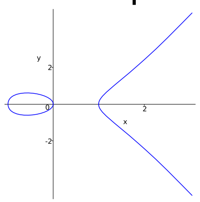
A cryptographic system is consider as sufficiently secure until someone finds an attack against it.

ELLIPTIC CURVES - PRELIMINARIES

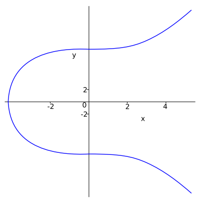
Elliptic curves E are graphs of points of plane curves defined by equations

$$E : y^2 = x^3 + ax + b,$$

For example:



$y^2 = x(x+1)(x-1)$



$y^2 = x^3 + 73$

Elliptic curves cryptography is based on a special operation of the addition of the points on elliptic curves at which it is easy to make addition of two points, but it is unfeasible to find first point given the sum of two points and second point.

ELLIPTIC CURVES CRYPTOGRAPHY and FACTORIZATION

- **Elliptic curve cryptography (ECC) is an approach to public-key cryptography based on the algebraic structure of points on elliptic curves over finite fields.**
- One of the main benefits of ECC, comparing with non-ECC, is that the same level of security is obtained using with ECC keys of the smaller size, than in case of other PKC such as RSA. That allows to reduce storage and transmission requirements.
- **For example, a 256-bit ECC public key should provide comparable security to a 3072-bit RSA public key.**
- ECC is used for encryption, digital signatures and PRG. Security of ECC is based on infeasibility of computing discrete logarithms of random elliptic curves elements.
- In USA two of the most prominent institutions, NIST (National Institute of Standards and Technology) and NSA (National Security Agency) endorsed in 2004 to use ECC with 384-bit keys for top secret communications.
- In August 2015 NSA announced plans to replace ECC cryptography by, not yet determined post-quantum cryptography.

COMMENTS I.

- **Elliptic curves belong to very important and deep mathematical concepts with a very broad use.**
- The use of elliptic curves for cryptography was suggested, independently, by Neal Koblitz and Victor Miller in 1985. ECC started to be widely used after 2005.
- **Elliptic curves are also the basis of a very important Lenstra's integer factorization algorithm.**
- Both of these uses of elliptic curves, ECC cryptography and ECC based integer factorization are dealt with in this chapter.

COMMENTS II.

- Elliptic curves are also seen by some mathematicians as the simplest non-trivial mathematical object.
- Historically, computing the integral of an arc-length of an ellipse lead to the idea of elliptic functions and curves.
- Niels Henrik Abel (1802-1829) and K. W. T. Weierstrass (1815-1897) are considered as pioneers in the area of elliptic functions.
- Abel has been considered, by his contemporaries, as mathematical genius that left enough for mathematicians to study for next 500 years.

COMMENTS III.

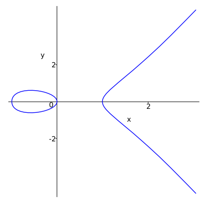
It is amazing how practical is the elliptic curve cryptography that is based on very strangely looking and very theoretical concepts.

ELLIPTIC CURVES

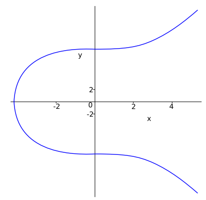
An elliptic curve E is the graph of points of the plane curve defined by the Weierstrass equation

$$E : y^2 = x^3 + ax + b,$$

(where a, b are either rational numbers or integers (and computation is then done modulo some integer n)) extended by a "point at infinity", denoted usually as ∞ (or 0) that can be regarded as being, at the same time, at the very top and very bottom of the y-axis.
We will consider only those elliptic curves that have no multiple roots - which is equivalent to the condition $4a^3 + 27b^2 \neq 0$.
In case coefficients and x, y can be any rational numbers, a graph of an elliptic curve has one of the forms shown in the following figure. The graph depends on whether the polynomial $x^3 + ax + b$ has three or only one real root.



$y^2 = x(x+1)(x-1)$



$y^2 = x^3 + 73$

MORE PRECISE DEFINITION

A more precise definition of elliptic curves requires that it is the curve of points of the equation

$$E : y^2 = x^3 + ax + b$$

in the case the curve is non-singular.

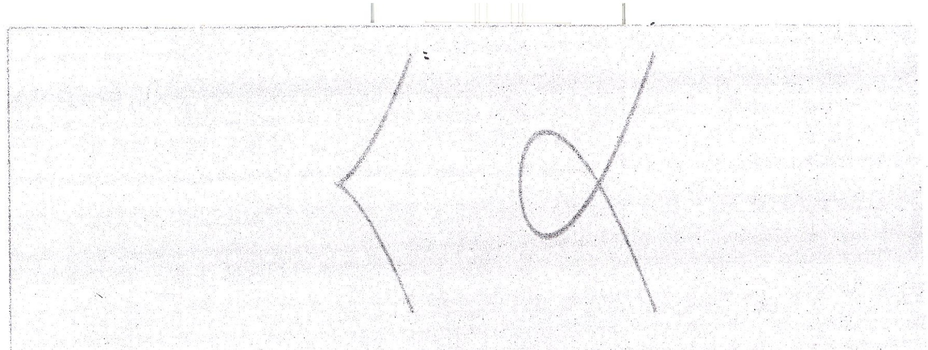
Geometrically, this means that the graph has no cusps, self-interactions, or isolated points.

Algebraically a curve is non-singular if and only if the discriminant

$$\Delta = -16(4a^3 + 27b^2) \neq 0$$

The graph of a non-singular curve has two components if its discriminant is positive, and one component if it is negative.

EXAMPLES OF SINGULAR "ELLIPTIC CURVES"



Types of singularities: on the left, a curve with a cusp ($y^2 = x^3$). On the right, a curve with a self-intersection ($y^2 = x^3 - 3x + 2$). None of them is a valid elliptic curve.

ADDITION of POINTS on ELLIPTIC CURVES - GEOMETRY

Geometry

On any elliptic curve we can define addition of points in such a way that points of the corresponding curve with such an operation of addition form an Abelian group in which the point in infinite, denoted by ∞, is plying the role of the identity group element.

If the line through two different points P_1 and P_2 of an elliptic curve E intersects E in a point $Q = (x, y)$, then we define $P_1 + P_2 = P_3 = (x, -y)$. (This also implies that for any point P on E it holds $P + \infty = P$.) ∞ therefore indeed play a role of the null/identity element of the group.

If the line through two different points P_1 and P_2 is parallel with y-axis, then we define $P_1 + P_2 = \infty$.

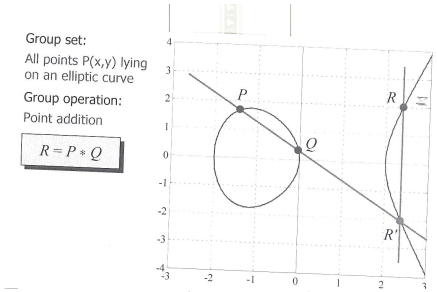
In case $P_1 = P_2$, and the tangent to E in P_1 intersects E in a point $Q = (x, y)$, then we define $P_1 + P_1 = (x, -y)$.

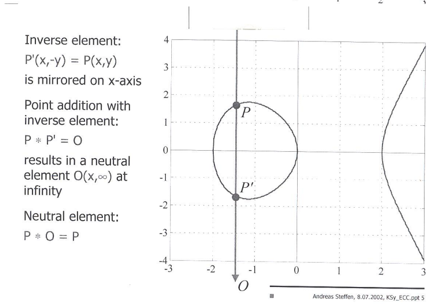
It should now be obvious how to define subtraction of two points of an elliptic curve.

It is now easy to verify that the above addition of points forms Abelian group with ∞ as the identity (null) element.

ADDITION of POINTS - EXAMPLES 1 and 2

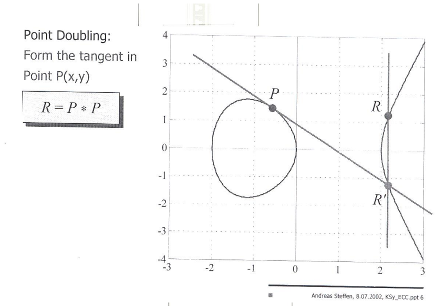
The following pictures show some cases of points additions

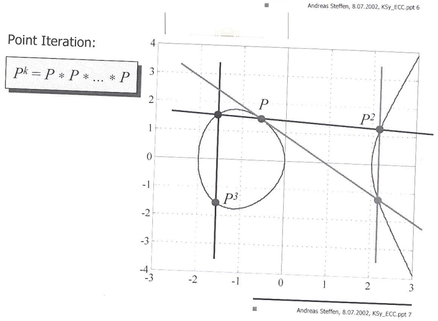




ADDITION of POINTS - EXAMPLES 3 and 4

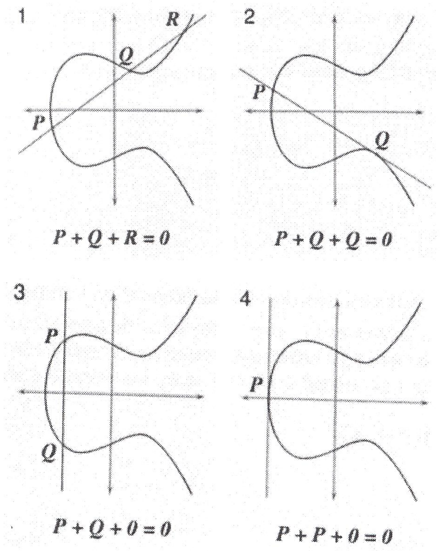
The following pictures show some cases of double and triple points additions





ADDITION of POINTS - EXAMPLES 5 and 6

The following pictures show some more complex cases of double and triple points additions



ADDITION of POINTS on ELLIPTIC CURVES - FORMULAS)

Formulas

Addition of points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ of an elliptic curve $E : y^2 = x^3 + ax + b$ can be easily computed using the following formulas:

$$P_1 + P_2 = P_3 = (x_3, y_3)$$

where

$$x_3 = \lambda^2 - x_1 - x_2$$
$$y_3 = \lambda(x_1 - x_3) - y_1$$

and

$$\lambda = \begin{cases} \frac{(y_2 - y_1)}{(x_2 - x_1)} & \text{if } P_1 \neq P_2, \\ \frac{(3x_1^2 + a)}{(2y_1)} & \text{if } P_1 = P_2. \end{cases}$$

All that holds for the case that $\lambda \neq \infty$; otherwise $P_3 = \infty$.
Example: For curve $y^2 = x^3 + 73$ and $P_1 = (2, 9)$, $P_2 = (3, 10)$ we have $\lambda = 1$, $P_1 + P_2 = P_3 = (-4, -3)$ and $P_3 + P_3 = (72, 611)$. – $\{\lambda = -8\}$

DERIVATION of FORMULAS for ADDITION of DIFFERENT POINTS

If $P_1 \neq P_2$, then the line that goes through points P_1 and P_2 has the equation

$$y = y_1 + \lambda(x - x_1) = y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x - x_1).$$

To get the x-coordinate of the third, intersection, point, of the curve $y^2 = x^3 + ax + b$ we have to find the third root of the equation:

$$y^2 = (y_1 + \lambda(x - x_1))^2 = x^3 + ax + b$$

that can be rewritten in the form

$$x^3 - \lambda^2 x^2 + (a - 2\lambda(y_1 - \lambda x_1))x + (b - (y_1 - \lambda x_1)^2) = 0$$

Since its two roots have coordinates x_1 and x_2 for the third, x_3, it has to hold

$$x_3 = \lambda^2 - (x_1 + x_2) = \lambda^2 - x_1 - x_2,$$

because $-\lambda^2$ is the coefficient at x^2 and therefore $x_1 + x_2 + x_3 = -(-\lambda^2) = \lambda^2$.

ELLIPTIC CURVES mod n

The points on an elliptic curve

$$E : y^2 = x^3 + ax + b \text{ (mod } n),$$

where a and b are integers, notation $E_n(a,b)$ are such pairs of integers (x,y), $|x| \leq n$, $|y| \leq n$, that satisfy the above equation, along with the point ∞ at infinity.

Example: Elliptic curve $E : y^2 = x^3 + 2x + 3$ (mod 5) has points

$$(1,1),(1,4),(2,0),(3,1),(3,4),(4,0),\infty.$$

Example For elliptic curve $E : y^2 = x^3 + x + 6($ mod 11) and its point $P = (2,7)$ it holds $2P = (5,2)$; $3P = (8,3)$. Number of points on an elliptic curve (mod p) can be easily estimated - as shown later.

The addition of points on an elliptic curve mod n is done by the same formulas as given previously, except that instead of rational numbers c/d we deal with cd^{-1} mod n

Example: For the curve $E : y^2 = x^3 + 2x + 3$ mod 5, it holds $(1,4)+(3,1) = (2,0); (1,4)+(2,0) = (?,?)$.

EXAMPLE OF AN ELLIPTIC CURVE OVER A PRIME

Points of the elliptic curve $y^2 = x^3 + x + 6$ over Z_{11}

x	$x^3 + x + 6$(mod 11)	in QR_{11}	y
0	6	no	
1	8	no	
2	5	yes	4,7
3	3	yes	5,6
4	8	no	
5	4	yes	2,9
6	8	no	
7	4	yes	2,9
8	9	yes	3,8
9	7	no	
10	4	yes	2,9

The number of points of an elliptic curve over Z_p is in the interval

$$(p + 1 - 2\sqrt{p}, p + 1 + 2\sqrt{p})$$

ADDITION of POINTS on ELLIPTIC CURVES - REPETITIONS

Formulas

Addition of points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ of an elliptic curve $E : y^2 = x^3 + ax + b$ can be easily computed using the following formulas:

$$P_1 + P_2 = P_3 = (x_3, y_3)$$

where

$$x_3 = \lambda^2 - x_1 - x_2$$
$$y_3 = \lambda(x_1 - x_3) - y_1$$

and

$$\lambda = \begin{cases} \frac{(y_2 - y_1)}{(x_2 - x_1)} & \text{if } P_1 \neq P_2, \\ \frac{(3x_1^2 + a)}{(2y_1)} & \text{if } P_1 = P_2. \end{cases}$$

All that holds for the case that $\lambda \neq \infty$; otherwise $P_3 = \infty$.

Example For curve $y^2 = x^3 + 73$ and $P_1 = (2,9)$, $P_2 = (3,10)$ we have $\lambda = 1$, $P_1 + P_2 = P_3 = (-4,-3)$ and $P_3 + P_3 = (72,611)$. – $\{\lambda = -8\}$

A VERY IMPORTANT OBSERVATION

In case of modular computation of coordinates of the sum of two points of an elliptic curve $E_n(a,b)$ one needs, in order to determine value of λ to compute $u^{-1}($ mod $n)$ for various u.

This can be done in case $gcd(u,n) = 1$ and therefore we need to compute $gcd(u,n)$ first.

Observe that if this gcd-value is between 1 and n we have a factor of n.

POINTS on CURVE $y^2 = x^3 + x + 6 \mod 11$

x	y^2	$y_{1,2}$	P(x,y)	P'(x,y)
0	6	-		
1	8	-		
2	5	4,7	(2,4)	(2,7)
3	3	5,6	(3,5)	(3,6)
4	8	-		
5	4	2,9	(5,2)	(5,9)
6	8	-		
7	4	2,9	(7,2)	(7,9)
8	9	3,8	(8,3)	(8,8)
9	7	-		
10	4	2,9	(10,2)	(10,9)

There are 12 points lying on the elliptic curve.

Together with the point O at infinity, the points on the elliptic curve form a group with n=13 elements.

n is called the order of the elliptic curve group and depends on the choice of the curve parameters a and b.

EXAMPLE

On the elliptic curve

$$y^2 \equiv x^3 + x + 6 \pmod{11}$$

lies the point $P = (2,7) = (x_1, y_1)$

Indeed, $49 \equiv 16 \mod 11$.

To compute $2P = (x_3, y_3)$ we have

$$\lambda = \frac{3x_1^2 + a}{2y_1} \equiv (3 \cdot 2^2 + 1)/(14) \equiv 13/14 \equiv 2/3 \equiv 2 \cdot 4 \equiv 8 \equiv \mod 11$$

Therefore

$$x_3 = \lambda^2 - x_1 - x_2 \equiv 8^2 - 2 - 2 \equiv 60 \equiv 5 \mod 11$$

and

$$y_3 = \lambda(x_1 - x_3) - y_1 \equiv 8(2-5) - 7 \equiv -31 \equiv -9 \equiv 2 \mod 11$$

PROPERTIES of ELLIPTIC CURVES MODULO p

- Elliptic curves modulo an integer p have finitely many points and are finitely generated - all points can be obtained from few given points using the operation of addition.
- Hasse's theorem If an elliptic curve E_p has $|E_p|$ points then $||E_p| - p - 1| < 2\sqrt{p}$

  In other words, the number of points of a curve grows roughly as the number of elements in the field. The exact number of such points is, however, rather difficult to calculate.

SECURITY of ECC

- The entire **security of ECC depends on** our **ability** to compute addition of two points and on **inability** to compute one summon given the sum and the second summon.
- However, no proof of security of ECC has been published so far.

USE OF ELLIPTIC CURVES IN CRYPTOGRAPHY

USE OF ELLIPTIC CURVES IN CRYPTOGRAPHY

ELLIPTIC CURVES DISCRETE LOGARITHM

Let E be an elliptic curve and A, B be its points such that $B = kA = (A + A + \ldots A + A)$ – k times – for some k. The task to find (given A and B) such a k is called the discrete logarithm problem for elliptic curves.

No efficient algorithm to compute discrete logarithm problem for elliptic curves is known and also no good general attacks. Elliptic curves based cryptography is based on these facts.

FROM DISCRETE LOGARITHM to ELLIPTIC CURVE DISCRETE LOGARITHMIC CRYPTO PROTOCOLS

There is the following general procedure for changing a discrete logarithm based crypto graphic protocols P to a crypto graphic protocols based on elliptic curves:

- Assign to a given message (plaintext) a point on the given elliptic curve E.
- Change, in the crypto graphic protocol P, modular multiplication to addition of points on E.
- Change, in the crypto graphic protocol P, each exponentiation to a multiplication of points of the elliptic curve E by integers.
- To the point of the elliptic curve E that results from such a protocol assign a message (cryptotext).

POWERS of POINTS

The following table shows powers of various points of the curve

$$y^2 = x^3 + x + 6 \mod 11$$

k	P^k	s	y_0
1	(2,4)	3	9
2	(5,9)	9	8
3	(8,8)	8	10
4	(10,9)	2	0
5	(3,5)	1	2
6	(7,2)	4	7
7	(7,9)	1	2
8	(3,6)	2	0
9	(10,2)	8	10
10	(8,3)	9	8
11	(5,2)	3	9
12	(2,7)	∞	-

Given an elliptic curve

$y^2 = x^3 + ax + b \mod p$

and a basis point P, we can compute

$Q = P^k$

through k-1 iterative point additions. Fast algorithms for this task exist. Unfortunately most of them are patented by Certicom and others.

Question: Is it possible to compute k when the point Q is known?

Answer: This is a hard problem known as the Elliptic Curve Discrete Logarithm.

where instead of λ an s is written.

MAPPING MESSAGES into POINTS of ELLIPTIC CURVES I.

Problem and basic idea

The problem of assigning messages to points on elliptic curves is difficult because there are no polynomial-time algorithms to write down points of an arbitrary elliptic curve.

Fortunately, there is a fast randomized algorithm, to assign points of any elliptic curve to messages, that can fail with probability that can be made arbitrarily small.

Basic idea: Given an elliptic curve $E(\bmod p)$, the problem is that not to every x there is an y such that (x, y) is a point of E.

Given a message (number) m we adjoin to m few bits at the end of m and adjust them until we get a number x such that $x^3 + ax + b$ is a square mod p.

EFFICIENCY of various CRYPTO GRAPHIC SYSTEMS

The following pictures show how many bits need keys of different crypto graphic systems to achieve the same security.

Equivalent Cryptographic Strength

Symmetric	56	80	112	128	192	256
RSA n	512	1024	2048	3072	7680	15360
ECC p	112	161	224	256	384	512
Key size ratio	5:1	6:1	9:1	12:1	20:1	30:1

ELLIPTIC CURVES KEY EXCHANGE

Elliptic curve version of the Diffie-Hellman key generation protocol goes as follows:

Let Alice and Bob agree on a prime p, on an elliptic curve $E_p(a, b)$ and on a point P on $E_p(a, b)$.

- Alice chooses an integer n_a, computes $n_A P$ and sends it to Bob.
- Bob chooses an integer n_b, computes $n_B P$ and sends it to Alice.
- Alice computes $n_a(n_B P)$ and Bob computes $n_b(n_A P)$. This way they have the same key.

ELLIPTIC CURVES VERSION of ElGamal CRYPTOSYSTEM

Standard version of ElGamal: Bob chooses a prime p, a generator $q < p$, an integer x, computes $y = q^x \pmod p$, makes public p, q, y and keeps x secret.

To send a message m Alice chooses a random r, computes:

$$a = q^r \ ; \ b = my^r$$

and sends it to Bob who decrypts by calculating $m = ba^{-x} \pmod p$

Elliptic curve version of ElGamal: Bob chooses a prime p, an elliptic curve E_p, a point P on E, an integer x, computes Q = xP, makes E_p, and Q public and keeps x secret.

To send a message m ALice expresses m as a point X on E_p, chooses a random number r, computes

$$A = rP \ ; \ B = X + rQ$$

and sends the pair (A, B) to Bob who decrypts by calculating $X = B - xA$.

COMMENT

There are two problems when implementing directly ElGamal cryptosystem on an elliptic curve:

- Expansion factor is 4;
- There is no deterministic method known to generate points (plaintexts) on the curve.

ELLIPTIC CURVES DIGITAL SIGNATURES

Elliptic curves version of ElGamal digital signatures has the following form for signing (a message) m, an integer, by Alice and to have the signature verified by Bob:

Alice chooses a prime p, an elliptic curve $E_p(a,b)$, a point P on E_p and calculates the number of points n on E_p – what can be done, and we assume that $0 < m < n$.

Alice then chooses a random integer a and computes Q = aP. She makes public p, E, P, Q and keeps secret a.

To sign a message m Alice does the following:

- Alice chooses a random integer $r, 1 \leq r < n$ such that gcd(r,n) = 1 and computes R = rP = (x,y).
- Alice computes $s = r^{-1}(m - ax) \pmod{n}$
- Alice sends the signed message (m,R,s) to Bob.

Bob verifies the signature as follows:

- Bob declares the signature as valid if xQ + sR = mP

  The verification procedure works because

$$xQ + sR = xaP + r^{-1}(m - ax)(rP) = xaP + (m - ax)P = mP$$

Warning Observe that actually $rr^{-1} = 1 + tn$ for some t. For the above verification procedure to work we then have to use the fact that $nP = \infty$ and therefore $P + t \cdot \infty = P$

COMMENT

Federal (USA) elliptic curve digital signature standard (ECDSA) was introduced in 2005.

Elliptic curve method was used to factor Fermat numbers F_{10} (308 digits) and F_{11} (610 digits).

DOMAIN PARAMETERS for ELLIPTIC CURVES

To use ECC, all parties involved have to agree on all basic elements concerning the elliptic curve E being used:

- A prime p.
- Constants a and b in the equation $y^2 = x^3 + ax + b$.
- Generator G of the underlying cyclic subgroup such that its order is a prime.
- The order n of G is the smallest integer n such that $nG = 0$
- Co-factor $h = \frac{|E|}{n}$ should be small ($h \leq 4$) and, preferably $h = 1$.

To determine domain parameters (especially n and h) may be much time consuming task. That is why mostly so called "standard or "named' elliptic curves are used that have been published by some standardization bodies.

SECURITY of ELLIPTIC CURVE CRYPTOGRAPHY

- Security of ECC depends on the difficulty of solving the discrete logarithm problem over elliptic curves.
- Two general methods of solving such discrete logarithm problems are known.
- The square root method and Silver-Pohling-Hellman (SPH) method.
- SPH method factors the order of a curve into small primes and solves the discrete logarithm problem as a combination of discrete logarithms for small numbers.
- Computation time of the square root method is proportional to $O(\sqrt{e^n})$ where n is the order of the based element of the curve.

KEY SIZE

- All known algorithms to solve elliptic curves discrete logarithm problem need at least $\theta(\sqrt{n})$ steps, where n is the order of the group.
- This implies that the size of the underlying field (number of points on the chosen elliptic curve) should be roughly twice the security parameter.
- For example, for 128-bit security one needs a curve over $\underline{F}_q$, where $q \approx 2^{256}$.
- This can be contrasted with RSA cryptography that requires 3072-bit public and private keys to keep the same level of security.

BREAKING ECC

- The hardest ECC scheme (publicly) broken to date had a 112-bit key for the prime field case and a 109-bit key for the binary field case.
- The prime field case was broken in July 2009 using 200 PlayStation 3 game consoles and could be finished in 3.5 months.
- The binary field case was broken in April 2004 using 2600 computers for 17 months.

GOOD ELLIPTIC CURVES

- NIST recommended 5 elliptic curves for prime fields, one for prime sizes 192, 224, 256, 384 and 521 bits.
- NIST also recommended five elliptic curves for binary fields $\mathbf{F}_{2^m}$ one for m equal 163, 233, 283, 409 and 571.

INTEGER FACTORIZATION

INTEGER FACTORIZATION

INTEGER FACTORIZATION - PROBLEM I

Two very basic questions concerning integers are of large theoretical and also practical cryptographical importance.

- **Can a given integer n be factorized? (Or, is n prime?)**
- **If n can be factorized, find its factors.**

Till around 1977 no polynomial algorithm was know to determine primality of integers. In spite of the fact that this problem bothered mathematicians since antique ancient times.

In 1977 **several very simple and fast randomized algorithms for primality testing were discovered** - one of them is on the next slide. One of them - Rabin-Miller algorithm - has already been discussed.
So called Fundamental theorem of arithmetic, known since Euclid, claims that factorization of an integer n into a power of primes

$$n = \prod_{i=1}^{k} p_i^{e_i}$$

is unique when primes p_i are ordered. However, theorem provides no clue how to find such a factorization and till now **no classical polynomial factorization algorithm is know.**

INTEGER FACTORIZATION - PROBLEM II

In 2002 a deterministic, so called ASK, polynomial time algorithm for primality testing, with complexity $O(n^{12})$ were discovered by three scientists from IIT Kanpur.

For factorization no polynomial deterministic algorithm is known and development of methods that would allow to factorized large integers is one of mega challenges for the development of computing algorithms and technology.

Largest recent success was factorization of so called RSA-768 number that has 232 digits (and 768 bits). Factorization took 2 years using several hundred of fast computers all over the world (using highly optimized implementation of the general field sieve method). On a single computer it would take 2000 years.

There is a lot of heuristics to factorized integers - some are very simple, other sophisticated. A method based on elliptic curves presented later, is one of them.

Factorization could be done in polynomial time using Shor's algorithm and a powerful quantum computer, as discussed later.

Fermat numbers FACTORIZATION

Factorization of so-called Fermat numbers $2^{2^i}+1$ is a good example to illustrate progress that has been made in the area of factorization.
Pierre de Fermat (1601-65) expected that all following numbers are primes:

$$F_i = 2^{2^i} + 1 \qquad i \geq 1$$

This is indeed true for $i = 0, \ldots, 4$. $F_0 = 3, F_1 = 5$, $F_2 = 17$, $F_3 = 257$, $F_4 = 65537$.

1732 L. Euler found that $F_5 = 4294967297 = 641 \cdot 6700417$
1880 Landry+LeLasser found that

$$F_6 = 18446744073709551617 = 274177 \cdot 67280421310721$$

1970 Morrison+Brillhart found factorization for $F_7 = (39 digits)$

$$F_7 = 340282366920938463463374607431768211457 =$$
$$= 5704689200685129054721 \cdot 59649589127497217$$

1980 Brent+Pollard found factorization for F_8
1990 A. K. Lenstra+ ... found factorization for F_9 (155 digits). Currently, also factorizations of F_{10} (308 digits) and F_{11} (610 digits) are known.

FACTORIZATION BASICS

- Not all numbers of a given length are equally hard to factor. The hardest instances are **semi-primes** - products of two primes of similar length.
- Concerning complexity classes it holds. **Function version of the factorization problem is known to be in FNP and it is not known to be in FP**.
- **Decision version of the factorization problem: Does an integer n has a factor smaller than d? is known to be in NP and not known to be in P.** Moreover it is known to be both in **NP** and **co-NP** as well both in **UP** and **co-UP**.
- The fastest known factorization algorithm has time

$$e^{(1.9 \ln n)^{1/3}(\ln \ln n)^{2/3})}$$

and with it we can factor 140 digit numbers in reasonable time.

BASIC FACTORIZATION METHODS.

BASIC FACTORIZATION METHODS

These methods are actually heuristics, and for each of them a variety of modifications is known.

TRIAL DIVISION

Algorithm Consider the list of all integers and a integer n to factorizeo. Divide n with all primes, 2, 3, 5, 7, 11,

13,.... up to $\sqrt{n}$ until you find a factor. If you do not find it n is prime,

Each time you divide **n** by a prime delete from the the list of considered integers all multiples of that prime.

Time complexity: $e^{\frac{1}{2}\ln n} = L(1, \frac{1}{2})$

Notation $L(\varepsilon, c)$ is used to denote complexity

$$O(e^{(c+o(1))(\ln n)^{\varepsilon}(\ln \ln n)^{1-\varepsilon}})$$

EULER's FACTORIZATION

The idea is to factorize an integer n by writing it at first as two different sums of two different integer squares. Famous example of Euler,

$$n = a^2 + b^2 = c^2 + d^2 - - - - - - - 1000009 = 1000^2 + 3^2 = 972^2 + 235^2$$

.
Denote then

$$k = gcd(a-c, d-b) \quad h = gcd(a+c, d+b)$$
$$m = gcd(a+c, d-b) \quad l = gcd(a-c, d+b)$$

In such a case either both k and h are even or both m and l are even. In the first case

$$n = ((\frac{k}{2})^2 + (\frac{h}{2})^2)(l^2 + m^2)$$

Unfortunately, disadvantage of Euler's factorization method is that it cannot be applied to factor an integer with any prime factor of the form $4k+3$ occurring to an odd power in its prime factorization.

FERMAT's FACTORIZATION

If n = pq, $p < \sqrt{n}$, then

$$n = \left(\frac{q+p}{2}\right)^2 - \left(\frac{q-p}{2}\right)^2 = a^2 - b^2$$

Therefore, in order to find a factor of n, we need only to investigate the values

$$x = a^2 - n$$

$$\text{for } a = \left\lceil\sqrt{n}\right\rceil + 1, \left\lceil\sqrt{n}\right\rceil + 2, \ldots, \frac{(n-1)}{2}$$

until a perfect square is found.

SIMPLE but POWER IDEAS

To find a factor of a given integer n do the following

- **Original idea: Generate, in a simple and clever way**, a pseudorandom sequence of integers x_0, x_1, x_2 and compute, for $i = 1, 2, \ldots gcd(x_i, n)$ until a factor of n is found.
- **Huge-computer-networks-era idea:** Generate, in a simple and clever way, huge number of well related pseudorandom sequences $x_0, x_1, \ldots$ and make a huge number of computers (all over the world) to compute, each for a portion of such sequences, $gcd(x_i, n)$ until one of them finds a factor of n.

Pollard ρ-FACTORIZATION - basic idea

To factorize an integer n:

1. Randomly choose $x_0 \in \{1, 2, \ldots, n\}$. Compute $x_i = x_{i-1}^2 + x_{i-1} + 1(\mod n)$, for $i = 1, 2, \ldots$

2. Two versions:
Version 1: Compute $gcd(x_i - x_j, n)$ for $i = 1, 2, \ldots$ and $j = 1, 2, \ldots, i-1$ until a factor of n is found.

Version 2: Compute $gcd(x_i - x_{2i}, n)$ for $i = 1, 2, \ldots$ until a factor is found.

Time complexity: $L(1, \frac{1}{4})$. **Note**: Some other polynomial than $x_{i-1}^2 + x_{i-1} + 1$ can be used.

The second method was used to factor 8-th Fermat number F_8 with 78 digits.

JUSTIFICATION of VERSION 1

Let p be a non-trivial factor of n much smaller than n.

Since there is a smaller number of congruence classes modulo p than modulo n, it is quite probable that there exist x_i and x_j such that

$$x_i \equiv x_j(\mod p) \text{ and } x_i \not\equiv x_j(\mod n)$$

In such a case $n \not| (x_i - x_j)$ and therefore $gcd(x_i - x_j, n)$ is a nontrivial factor of n.

JUSTIFICATION of VERSION 2

Let p be the smallest factor of n.
Sequence $x_0, x_1, x_2, \ldots.$ behaves randomly modulo $p \leq \sqrt{n}$.
Therefore, the probability that $x_i \equiv x_j (\mod p)$ for some $j \neq i$ is not negligible - actually about $\frac{1}{\sqrt{p}}$.
In such a case $x_{i+k} \equiv x_{j+k} (\mod p)$ for all k
Therefore, there exists an s such that $x_s \equiv x_{2s} (\mod p)$.
Due to the pseudorandomness of the sequence x_0, x_1, x_2, with probability at least $1/2$ $x_s \not\equiv x_{2s} (\mod n)$ and therefore $p | gcd(x_s - x_{2s}, n)$.
For good probability of success we need to generate roughly $\sqrt{p} = n^{1/4}$ of x_i. Time complexity is therefore $O(e^{\frac{1}{4}\ln n})$.

BASIC FACTS

Factorization using ρ-algorithms has its efficiency based on two facts.

- **Fact 1** For a given prime p, as in birthday problem, two numbers are congruent modulo p, with probability 0.5 after $1.177\sqrt{p}$ numbers have been randomly chosen.
- **Fact 2** If p is a factor of an n, then $p < gcd(x - y, n)$ since p divides both n and $x - y$.

ρ-ALGORITHM - EXAMPLE

$$f(x) = x^2 + x + 1$$
$$n = 18923; \quad x = y = x_0 = 2347$$
$$x \leftarrow f(x) \bmod n; y \leftarrow f(f(y)) \bmod n$$
$$gcd(x - y, n) = ?$$

x	=	4164	y	=	9593	$gcd(x-y,n)$	=	1
x	=	9593	y	=	2063	gcd	=	1
x	=	12694	y	=	14985	gcd	=	1
x	=	2063	y	=	14862	gcd	=	1
x	=	358	y	=	3231	gcd	=	1
x	=	14985	y	=	3772	gcd	=	1
x	=	5970	y	=	16748	gcd	=	1
x	=	14862	y	=	3586	gcd	=	1
x	=	5728	y	=	16158	gcd	=	149

Pollard $p - 1$ ALGORITHM - FIRST VERSION

Algorithm

To find a prime factor p.

1. Fix an integer B.
2. Compute $m = \prod_{\{q \,|\, q \text{ is a prime} \leq B\}} q^{\log n}$
3. Compute $gcd(a^m - 1, n)$ for a random a.

Algorithm was invented J. Pollard in 1987 and has time complexity $O(B(\log n)^p)$. It works well if both $p|n$ and $p - 1$ have only small prime factors.

JUSTIFICATION of FIRST Pollard's $p-1$ ALGORITHM

Let a bound B be chosen and let $p|n$ and $p-1$ has no factor greater than B.

This implies that $(p-1)|m$, where

$$m = \prod_{\{q \mid q \text{ is a prime} \leq B\}} q^{\log B}$$

By Fermat's Little Theorem, this implies that $p|(a^m - 1)$ for any integer a and therefore by computing

$$gcd(a^m - 1, n)$$

(for some a) some factor p of n can be obtained.

FACTORING with ELLIPTIC CURVES

Basis idea: To factorize an integer n choose an elliptic curve E_n, a point P on E and compute, modulo n, either iP for $i = 2, 3, 4, \ldots$ or $2^j P$ for $j = 1, 2, \ldots$.

The point is that in such calculations one needs to compute gcd(k,n) for various k. If one of these values is > 1 a factor of n is found.

Factoring of large integers: The above idea can be easily parallelised and converted to using an enormous number of computers to factor a single very large n.

Indeed, each computer gets some number of elliptic curves and some points on them and multiplies these points by some integers according to the rule for addition of points. If one of computers encounters, during such a computation, a need to compute $1 < gcd(k, n) < n$, factorization is finished.

Example: If curve $E : y^2 = x^3 + 4x + 4 \pmod{2773}$ and its point $P = (1, 3)$ are used, then $2P = (1771, 705)$ and in order to compute 3P one has to compute $gcd(1770, 2773) = 59$ – factorization is done.

A BRIEF VERSION of THE BASIC ALGORITHM

1. Fix a B - to be a factor base (of all primes smaller than B).
2. Compute
$$m = \prod_{\{q \mid q \text{ is a prime} \leq B\}} q^{\log B}.$$
3. Choose random a, b such that $a^3 - 27b^2 \neq 0$ (mod n).
4. Choose randomly a point P on the elliptic curve $E_n(a, b)$.
5. Try to compute mP.

EXAMPLE

Example: For elliptic curve

$$E : y^2 = x^3 + x - 1 \pmod{35}$$

and its point $P = (1, 1)$ we have

$$2P = (2, 32); 4P = (25, 12); 8P = (6, 9)$$

and at the attempt to compute 9P one needs to compute $gcd(15, 35) = 5$ and factorization is done.

It remains to be explored how efficient this method is and when it is more efficient than other methods.

IMPORTANT OBSERVATIONS (1)

- If $n = pq$ for primes p, q, then an elliptic curve E_n can be seen as a pair of elliptic curves E_p and E_q.
- It follows from the Lagrange theorem that for any elliptic curve E_n and its point P there is an $k < n$ such that $kP = \infty$.
- In case of an elliptic curve E_p for some prime p, the smallest positive integer m such that $mP = \infty$ for some point P divides the number N_p of points on the curve E_p. Hence $N_pP = \infty$.

  If N is a product of small primes, then b! will be a multiple of N for a reasonable small b. Therefore, $b!P = \infty$.
- The number with only small factors is called smooth and if all prime factors are smaller than an b, then it is called b-smooth.

It can be shown that the density of smooth integers is so large that if we choose a random elliptic curve E_n then it is a reasonable chance that n is smooth.

PRACTICALITY of FACTORING USING ECC I

Let us continue to discuss the following key problem for factorization using elliptic curves:

Problem: How to choose an integer k such that for a given point P we should try to compute points iP or 2^iP for all multiples of P smaller than kP?

Idea: If one searches for m-digits factors, one chooses k in such a way that k is a multiple of as many as possible of those m-digit numbers which do not have too large prime factors. In such a case one has a good chance that k is a multiple of the number of elements of the group of points of the elliptic curve modulo n.

Method 1: One chooses an integer B and takes as k the product of all maximal powers of primes smaller than B.

Example: In order to find a 6-digit factor one chooses B=147 and $k = 2^7 \cdot 3^4 \cdot 5^3 \cdot 7^2 \cdot 11^2 \cdot 13 \cdot \ldots \cdot 139$. The following table shows B and the number of elliptic curves one has to test:

PRACTICALITY of FACTORING USING ECC - II

Digits of to-be-factors	6	9	12	18	24
B	147	682	2462	23462	162730
Number of curves	10	24	55	231	833

Computation time by the elliptic curves method depends on the size of factors.

ELLIPTIC CURVES FACTORIZATION: FAQ

- How to choose (randomly) an elliptic curve E and point P on E? An easy way is first choose a point $P(x, y)$ and an a and then compute $b = y^2 - x^3 - ax$ to get the curve $E : y^2 = x^3 + ax + b$.
- What happens at the factorization using elliptic curve method, if for a chosen curve E_n the corresponding cubic polynomial $x^3 + ax + b$ has multiple roots (that is if $4a^3 + 27b^2 = 0$) ? No problem, method still works.
- What kind of elliptic curves are really used in cryptography? Elliptic curves over fields $GF(2^n)$ for $n > 150$. Dealing with such elliptic curves requires, however, slightly different rules.
- History of ECC? The idea came from Neal Koblitz and Victor S. Miller in 1985. Best known algorithm is due to Lenstra.
- How secure is ECC? No mathematical proof of security is know.
- How about patents concerning ECC? There are patents in force covering certain aspects of ECC technology.

FACTORIZATION on QUANTUM COMPUTERS

In the following we present the basic idea behind a polynomial time algorithm for quantum computers to factorize integers.

Quantum computers works with superpositions of basic quantum states on which very special (unitary) operations are applied and and very special quantum features (non-locality) are used.

Quantum computers work not with bits, that can take on any of two values 0 and 1, but with qubits (quantum bits) that can take on any of infinitely many states $\alpha|0\rangle + \beta|1\rangle$, where α and β are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$.

REDUCTIONS

Shor's polynomial time quantum factorization algorithm is based on an understanding that factorization problem can be reduced

1. first on the problem of solving a simple modular quadratic equation;
2. second on the problem of finding periods of functions $f(x) = a^x \bmod n$.

FIRST REDUCTION

Lemma If there is a polynomial time deterministic (randomized) [quantum] algorithm to find a nontrivial solution of the modular quadratic equations

$$a^2 \equiv 1 \pmod{n},$$

then there is a polynomial time deterministic (randomized) [quantum] algorithm to factorize integers.

Proof. Let $a \neq \pm 1$ be such that $a^2 \equiv 1 \pmod{n}$. Since

$$a^2 - 1 = (a+1)(a-1),$$

if n is not prime, then a prime factor of n has to be a prime factor of either $a+1$ or $a-1$.

By using Euclid's algorithm to compute

$$gcd(a+1, n) \quad \text{and} \quad gcd(a-1, n)$$

we can find, in $O(\lg n)$ steps, a prime factor of n.

SECOND REDUCTION

The second key concept is that of the **period** of functions

$$f_{n,x}(k) = x^k \bmod n.$$

Period is the smallest integer r such that

$$f_{n,x}(k+r) = f_{n,x}(k)$$

for any k, i.e. the smallest r such that

$$x^r \equiv 1 \pmod{n}.$$

AN ALGORITHM TO SOLVE EQUATION $x^2 \equiv 1 \pmod{n}$.

1. *Choose randomly $1 < a < n$.*
2. *Compute $gcd(a, n)$. If $gcd(a, n) \neq 1$ we have a factor.*
3. *Find period r of function $a^k \bmod n$.*
4. *If r is odd or $a^{r/2} \equiv \pm 1 \pmod{n}$, then go to step 1; otherwise stop.*

If this algorithm stops, then $a^{r/2}$ is a non-trivial solution of the equation

$$x^2 \equiv 1 \pmod{n}.$$

EXAMPLE

Let $n = 15$. Select $a < 15$ such that $gcd(a, 15) = 1$.
$\{$The set of such a is $\{2, 4, 7, 8, 11, 13, 14\}\}$

Choose $a = 11$. Values of $11^x \bmod 15$ are then

$$11, 1, 11, 1, 11, 1$$

which gives $r = 2$.

Hence $a^{r/2} = 11 \pmod{15}$. Therefore

$$gcd(15, 12) = 3, \qquad gcd(15, 10) = 5$$

For $a = 14$ we get again $r = 2$, but in this case

$$14^{2/2} \equiv -1 \pmod{15}$$

and the following algorithm fails.

1. *Choose randomly* $1 < a < n$.
2. *Compute* $gcd(a, n)$. *If* $gcd(a, n) \neq 1$ *we have a factor.*
3. *Find period* r *of function* $a^k \bmod n$.
4. *If* r *is odd or* $a^{r/2} \equiv \pm 1 \pmod n$,*then go to step 1; otherwise stop.*

EFFICIENCY of REDUCTION

Lemma If $1 < a < n$ satisfying $gcd(n, a) = 1$ is selected in the above algorithm randomly and n is not a power of prime, then

$$Pr\{r \text{ is even and } a^{r/2} \not\equiv \pm 1\} \geq \frac{9}{16}.$$

1. *Choose randomly* $1 < a < n$.
2. *Compute* $gcd(a, n)$. *If* $gcd(a, n) \neq 1$ *we have a factor.*
3. *Find period* r *of function* $a^k \bmod n$.
4. *If* r *is odd or* $a^{r/2} \equiv \pm 1 \pmod n$,*then go to step 1; otherwise stop.*

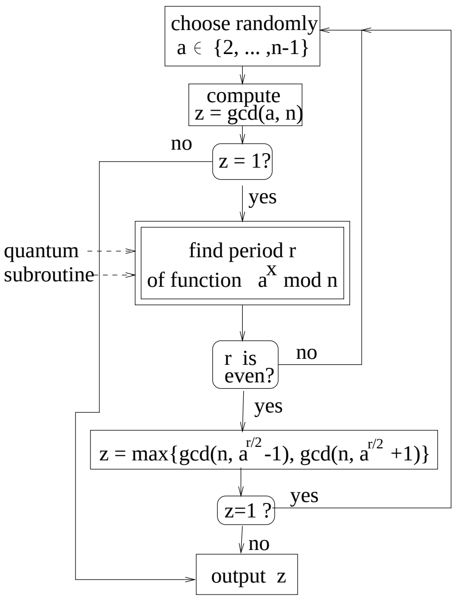
Corollary If there is a polynomial time randomized [quantum] algorithm to compute the period of the function

$$f_{n,a}(k) = a^k \bmod n,$$

then there is a polynomial time randomized [quantum] algorithm to find non-trivial solution of the equation $a^2 \equiv 1 \pmod n$ (and therefore also to factorize integers).

A GENERAL SCHEME for Shor's ALGORITHM

The following flow diagram shows the general scheme of Shor's quantum factorization algorithm



QUADRATIC SIEVE METHOD of FACTORIZATION - BASIC IDEAS

Step 1 To factorize an n one finds many integers x such that $x^2 - n$ has only small factors and decomposition of $x^2 - n$ into small factors.

Example - $n = 7429$

$$\left.\begin{array}{l} 83^2 - 7429 = -540 = (-1) \cdot 2^2 \cdot 3^3 \cdot 5 \\ 87^2 - 7429 = 140 = 2^2 \cdot 5 \cdot 7 \\ 88^2 - 7429 = 315 = 3^2 \cdot 5 \cdot 7 \end{array}\right\} \text{relations}$$

Step 2 One multiplies some of the relations such that their product is a square. For example

$$(87^2 - 7429)(88^2 - 7429) = 2^2 \cdot 3^2 \cdot 5^2 \cdot 7^2 = 210^2$$

Now, compute product modulo n:

$$(87^2 - 7429)(88^2 - 7429) \equiv (87 \cdot 88)^2 = 7656^2 \equiv 227^2 \mod 7429$$
$$\text{and therefore } 227^2 \equiv 210^2 \mod 7429$$

Hence 7429 divides $227^2 - 210^2$ and therefore $17 = 227 - 210$ is a factor of 7429.

A method to choose relations to form equations: For the i-th relation one takes a variable λ_i and forms the expression

$$((-1) \cdot 2^2 \cdot 3^3 \cdot 5)^{\lambda_1} \cdot (2^2 \cdot 5 \cdot 7)^{\lambda_2} \cdot (3^2 \cdot 5 \cdot 7)^{\lambda_3} = (-1)^{\lambda_1} \cdot 2^{2\lambda_1 + 2\lambda_2} \cdot 3^{2\lambda_1 + 2\lambda_2} \cdot 5^{\lambda_1 + \lambda_2 + \lambda_3} \cdot 7^{\lambda_2 + \lambda_3}$$

If this is to form a square the following equations have to hold

$$\begin{array}{ll} \lambda_1 & \equiv 0 \mod 2 \\ \lambda_1 + \lambda_2 + \lambda_3 & \equiv 0 \mod 2 \\ \lambda_2 + \lambda_3 & \equiv 0 \mod 2 \end{array}$$

Therefore: $\lambda_1 = 0, \quad \lambda_2 = \lambda_3 = 1$

QUADRATIC SIEVE FACTORIZATION - SKETCH of METHODS

Problem How to find which relations to choose?
Using the algorithm called Quadratic sieve method.

Step 1 One chooses a set of primes that can be factors – a so-called factor basis.

One chooses an m such that $m^2 - n$ is small and considers numbers $(m+u)^2 - n$ for $-k \leq u \leq k$ for small k.
One then tries to factor all $(m+u)^2 - n$ with primes from the factor basis, from the smallest to the largest - see table for n=7429 and m=86.

u	-3	-2	-1	0	1	2	3
$(m+u)^2 - n$	-540	-373	-204	-33	140	315	492
Sieve with 2	-135		-51		35		123
Sieve with 3	-5		-17	-11		35	41
Sieve with 5	-1				7	7	
Sieve with 7					1	1	

In order to factor a 129-digit number from the RSA challenge they used

8 424 486 relations
569 466 equations
544 939 elements in the factor base

QUADRATIC SIEVE (QS) FACTORIZATION - SUMMARY I

- Method was invented Carl Pomerance in 1981.
- It is currently second fastest factorization method known and the fastest one for factoring integers under 100 decimal digits.
- It consists of two phases: data collection and data processing.
- In data collection phase for factoring n a huge set of such integers x is found that numbers $(x + \lceil\sqrt{n}\rceil)^2 - n$ have only small factors as well all these factors. This phase is easy to parallelize and can use methods called **sieving** for finding all required integers with only small factors.
- In data processing phase a system of linear congruences is formed on the basis of factorizations obtained in the data collection phase and this system is solved to reach factorization. This phase is much memory consuming for storing huge matrices and so hard to parallelise.
- The basis of sieving is the fact that if $y(x) = x^2 - n$, then for any prime p it holds $y(x+kp) \equiv y(x) \pmod{p}$ and therefore solving $y(x) \equiv 0 \pmod{p}$ for x generate a whole sequence of y which are divisible by p.
- The general running time of QS, to factor n, is

$$e^{(1+o(1))\sqrt{\lg n \lg \lg n}}$$

- The current record of QS is a 135-digit co-factor of $2^{803} - 2^{402} - 1$.

ELLIPTIC CURVES FACTORIZATION - DETAILS

Given an n such that gcd(n, 6) = 1 and let the smallest factor of n be expected to be smaller than an F. One should then proceed as follows:
Choose an integer parameter **r** and:

1. Select, randomly, an elliptic curve
$$E : y^2 = x^3 + ax + b$$
such that $gcd(n, 4a^2 + 27b^2) = 1$ and a random point P on E.
2. Choose integer bounds A,B,M such that
$$M = \prod_{j=1}^{l} p_j^{a_j}$$
for some primes $p_1 < p_2 < \ldots < p_l \leq B$ and a_j, being the largest exponent such that $p_j^{a_j} \leq A$.
Set j = k = 1
3. Calculate $p_j P$.
4. Computing gcd.
   - If $p_j P \neq O \pmod{n}$, then set $P = p_j P$ and reset $k \leftarrow k+1$
     1. If $k \leq a_{p_j}$, then go to step (3).

ELLIPTIC CURVES FACTORIZATION - DETAILS II

2. If $k > a_j$, then reset $j \leftarrow j+1$, $k \leftarrow 1$.
   If $j \leq l$, then go to step (3); otherwise go to step (5)

- If $p_j P \equiv O(\mod n)$ and no factor of n was found at the computation of inverse elements, then go to step (5)

5. Reset $r \leftarrow r-1$. If $r > 0$ go to step (1); otherwise terminate with "failure".
The "smoothness bound" B is recommended to be chosen as
$$B = e^{\sqrt{\frac{lnF(lnlnF)}{2}}}$$
and in such a case the running time is
$$O(e^{\sqrt{2+o(1 lnF(lnlnF))} ln^2 n})$$

FACTORING ALGORITHMS RUNNING TIMES

Let p denote the smallest factor of an integer n and p^* the largest prime factor of $p-1$.

1. Pollard's Rho algorithm $O(\sqrt{p})$
2. Pollard's $p-1$ algorithm $O(p^*)$
3. Elliptic curve method $\emptyset(e^{(1+o(1))\sqrt{2 \ln p \ln \ln p)}})$
4. Quadratic sieve method $\emptyset(e^{1+o(1))\sqrt{(\ln n \ln \ln n)}})$
5. General number field sieve (GNFS) method $\emptyset(e^{(\frac{64}{9} \ln n)^{1/3}(\ln \ln n)^{2/3}})$

The most efficient factorization method, for factorization of integers with more than 100 digits, is the general number field sieve method (superpolynomial but sub-exponential); The second fastest is the quadratic sieve method.

APPENDIX

APPENDIX

HISTORICAL REMARKS on ELLIPTIC CURVES

Elliptic curves are not ellipses and therefore it seems strange that they have such a name. Elliptic curves actually received their names from their relation to so called elliptic integrals

$$\int_{x1}^{x2} \frac{dx}{\sqrt{x^3 + ax + b}} \qquad \int_{x1}^{x2} \frac{xdx}{\sqrt{x^3 + ax + b}}$$

that arise in the computation of the arc-length of ellipses.

It may also seem puzzling why to consider curves given by equations

$$E : y^2 = x^3 + ax + b$$

and not curves given by more general equations

$$y^2 + cxy + dy = x^3 + ex^2 + ax + b$$

The reason is that if we are working with rational coefficients or mod p, where $p > 3$ is a prime, then such a general equation can be transformed to our special case of equation. In other cases, it may be indeed necessary to consider the most general form of equation.

ELLIPTIC CURVES - GENERALITY

A general elliptic curve over Z_{p^m} where p is a prime is the set of points (x, y) satisfying so-called Weierstrass equation

$$y^2 + uxy + vy = x^3 + ax^2 + bx + c$$

for some constants u, v, a, b, c together with a single element $\mathbf{0}$, called the point of infinity.

If $p \neq 2$ Weierstrass equation can be simplified by transformation

$$y \rightarrow \frac{y - (ux + v)}{2}$$

to get the equation

$$y^2 = x^3 + dx^2 + ex + f$$

for some constants d, e, f and if $p \neq 3$ by transformation

$$x \rightarrow x - \frac{d}{3}$$

to get equation

$$y^2 = x^3 + fx + g$$

HISTORY of ELLIPTIC CURVES CRYPTOGRAPHY

- The use of elliptic curves in cryptography was suggested independently by Neal Koblitz and Victor S. Miller in 1985.
- Behind this method is the belief that the discrete logarithm of a random elliptic curve element with respect to publicly known base point is infeasible.
- At first only elliptic curves over a prime finite field were used for ECC. Later also elliptic curves over the fields $GF(2^m)$ started to be used.
- In 2005 the US NSA endorsed to use ECC (Elliptic curves cryptography) with 384-bit key to protect information classified as "top secret".
- There are patents in force covering certain aspects of ECC technology.
-
- Elliptic curves have been first used for factorization by Lenstra.
- Elliptic curves played an important role in perhaps most celebrated mathematical proof of the last hundred years - in the proof of Fermat's Last Theorem - due to A. Wiles and R. Taylor.

ELLIPTIC CURVES FACTORIZATION - DETAILS

Given an n such that gcd(n, 6) = 1 and let the smallest factor of n be expected to be smaller than an F. One should then proceed as follows:
Choose an integer parameter r and:

1. Select, randomly, an elliptic curve
$$E : y^2 = x^3 + ax + b$$
such that $gcd(n, 4a^2 + 27b^2) = 1$ and a random point P on E.
2. Choose integer bounds A,B,M such that
$$M = \prod_{j=1}^{l} p_j^{a_{p_j}}$$
for some primes $p_1 < p_2 < \ldots < p_l \leq B$ and a_{p_j}, being the largest exponent such that $p_j^{a_j} \leq A$.
Set j = k = 1
3. Calculate $p_j P$.
4. Computing gcd.
   - If $p_j P \neq O \pmod n$, then set $P = p_j P$ and reset $k \leftarrow k + 1$
     1. If $k \leq a_{p_j}$, then go to step (3).

ELLIPTIC CURVES FACTORIZATION - DETAILS II

2. If $k > a_{p_j}$, then reset $j \leftarrow j + 1$, $k \leftarrow 1$.
If $j \leq l$, then go to step (3); otherwise go to step (5)

- If $p_j P \equiv O(\mod n)$ and no factor of n was found at the computation of inverse elements, then go to step (5)

5. Reset $r \leftarrow r - 1$. If $r > 0$ go to step (1); otherwise terminate with "failure".
The "smoothness bound" B is recommended to be chosen as
$$B = e^{\sqrt{\frac{lnF(lnlnF)}{2}}}$$
and in such a case running time is
$$O(e^{\sqrt{2 + o(1 lnF(lnlnF))ln^2 n}})$$

KEY SIZE

- All fastest known algorithms to solve elliptic curves discrete logarithm problem need $O(\sqrt{n})$ steps.
- This implies that the size of the underlying field (number of points on the chosen elliptic curve) should be roughly twice the security parameter.
- For example, for 128-bit security one needs a curve over $\underline{F}_q$, where $q \approx 2^{256}$.
- This can be contrasted with RSA cryptography that requires 3072 public and private keys.

SECURITY of ELLIPTIC CURVE CRYPTOGRAPHY

- Security of ECC depends on the difficulty of solving the discrete logarithm problem over elliptic curves.
- Two general methods of solving such discrete logarithm problems are known.
- The square root method and Silver-Pohling-Hellman (SPH) method.
- SPH method factors the order of a curve into small primes and solves the discrete logarithm problem as a combination of discrete logarithms for small numbers.
- Computation time of the square root method is proportional to $O(\sqrt{e^n})$ where n is the order of the based element of the curve.

BREAKING ECC

- The hardest ECC scheme (publicly) broken to date had a 112-bit key for the prime field case and a 109-bit key for the binary field case.
- The prime field case was broken in July 2009 using 200 PlayStation 3 game consoles and could be finished in 3.5 months.
- The binary field case was broken in April 2004 using 2600 computers for 17 months.

GOOD ELLIPTIC CURVES

- NIST recommended 5 elliptic curves for prime fields, one for prime sizes 192, 224, 256, 384 and 521 bits.
- NIST also recommended five elliptic curves for binary fields $\mathbf{F}_{2^m}$ one for m equal 163, 233, 283, 409 and 571.